VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2017 Jan Kolářík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Kolářík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: GX SOLUTIONS Bohemia s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Lukáš**

Konzultant bakalářské práce: Ing. Marek Bober

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 1. dubna 2017 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 1. dubna 2017

.....

Rád bych na tomto místě poděkoval především Ing. Marku Boberovi za vedení a cenné rady, které vedly k úspěšnému absolvování mé odborné praxe, a také Ing. Petru Lukášovi za konzultace a korekturu potřebnou k doladění této práce.

Abstrakt

V této bakalářské práci je rozebrána odborná praxe absolvovaná ve společnosti GX Solutions Bohemia s.r.o. (dále jen GX Solutions). Práce je zaměřená na popis implementace nadstaveb monitorovacího systému společnosti GX Solutions. V úvodu je představena samotná společnost GX Solutions a pracovní zařazení studenta během jeho odborné praxe. Dále je popsaná architektura systému, na kterém student pracoval, a také technologie použité při implementaci. Následně jsou stručně popsány zadání úkolů, které byly studentovi zadány a následně jsou podrobně rozepsány postupy řešení těchto úkolů. Závěrem jsou rozepsány znalosti ze studia, které student uplatnil i chybějící znalosti, které musel získat samostatně.

Klíčová slova: bakalářská praxe, MVC, WCF, Entity Framework, DevExpress, PostgreSQL, routing, GX Solutions

Abstract

This thesis is contains the details of the individial professinal practice in the company GX Solutions Bohemia s.r.o. (hereinafter GX Solutions). The thesis is focused on describing the implementation of extensions to the monitoring system of GX Solutions.The introduction starts with the company GX Solutions itself and continues with work assignment of the student through his professional practice. Subsequently there are details of the architecture of the system which the student has worked on. Subsequently there is a brief description of the student's tasks and then the detailed description of the implemetation details of these tasks. In conclusion there is a summary of the knowledge which the student used, and which he had to learn individually

Key Words: bechelor thesis, MVC, WCF, Entity Framework, DevExpress, PostgreSQL, routing, GX Solutions

Obsah

Seznam použitých zkratek a symbolů

ASP	–	Active Server Pages
MVC	–	Model-View-Controller
WCF	–	Windows Communication Foundation
HTML	–	Hyper Text Markup Language
SQL	–	Structured Query Language
ACID	–	Atomicity, Consistency, Isolation, Durability
LINQ	–	Language Integrated Query
ORM	–	Objektově relační mapování
JS	–	JavaScript
IT	–	Informační technologie
IS	–	Informační systém
OS	–	Operační systém
API	–	Application Programming Interface
AJAX	–	Asynchronous JavaScript and XML
JSON	–	JavaScript Object Notation
XML	–	Extensible Markup Language
KML	–	Keyhole Markup Language
GUI	–	Graphical user interface
DTO	–	Data transfer object
CRUD	–	Create, read, update and delete

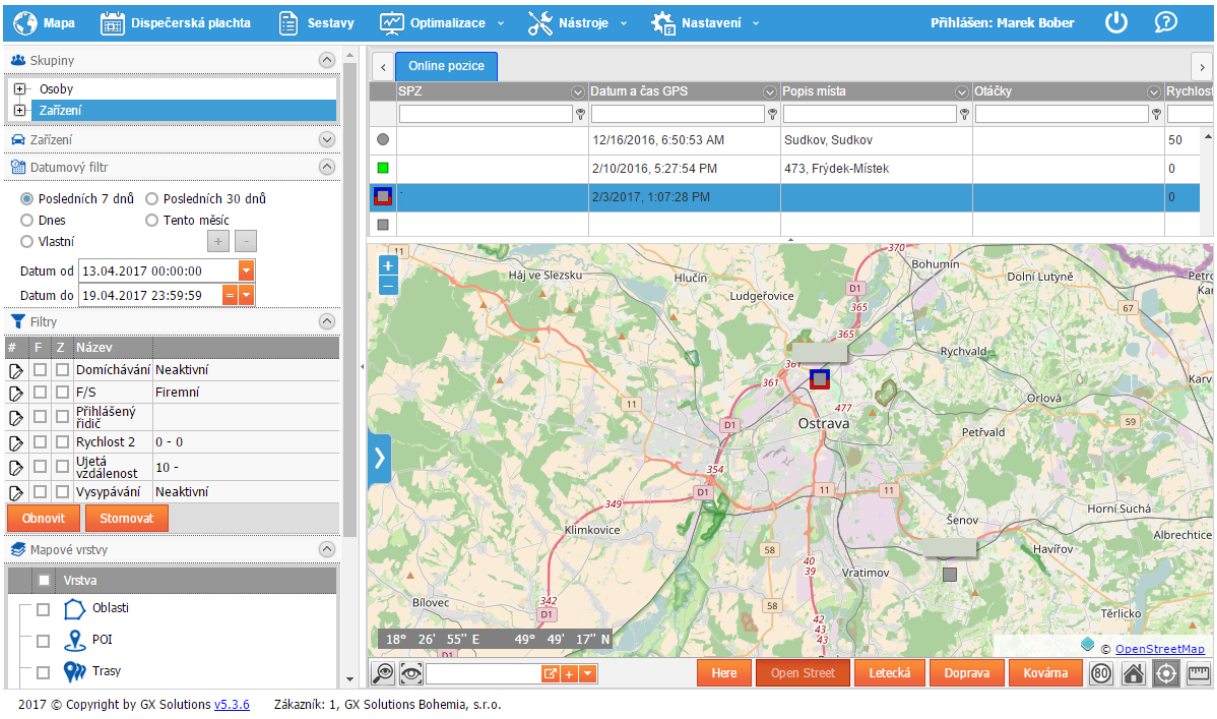
Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

GX Solutions je společnost založená v roce 1998. Jejím hlavním zaměřením jsou telematická řešení firmám s jakýmkoliv vozovým parkem. Nejdříve poskytovala informace o využívání a stavu vozidel především u kamiónů. V současnosti jsou její služby určeny pro vozové parky různého typu, složení a účelu užívání. Orientuje se na sledování aut, strojů, techniky a dalších zařízení v různých oblastech a odvětvích. Jde o to mezinárodní společnost působící v Česku a Slovensku a svými aktivitami roste i v Polsku, Maďarsku a Německu.[1]

Během své odborné praxe jsem byl přidělen do IT oddělení ve Frýdku-Místku spadající pod dceřinou společnost GX Innovation s.r.o. na pozici programátora. Primárním zaměřením mé práce byla implementace nadstaveb nad modulem sloužícím k výpočtům tras (dále jen modul routingu) ve webové aplikaci GX GO (viz obr. 1) vyvíjené touto společností.



Obrázek 1: Uživatelské rozhraní aplikace GX GO

2 Použité technologie

2.1 .NET Framework

.NET Framework je rozsáhlá softwarová platforma od společnosti Microsoft, která je určena pro vývoj mnoha různých druhů aplikací. Za pomoci .NET Frameworku můžeme vyvíjet nejen desktopové aplikace pro Windows, ale mimo jiné i webové aplikace a služby, aplikace pro mobilní zařízení a mnoho dalších.[2]

2.2 WCF

Windows Communication Foundation je sada knihoven, API a běhového prostředí, dohromady tvořící jednotný programovací model od společnosti Microsoft, v rámci frameworku .NET. Byl vyvinut pro budování aplikací orientovaných na služby.[3]

2.3 ASP.NET MVC

Jedná se o webový rámec, který je součástí technologie .NET a implementuje vzor MVC (Model-View-Controller). Tato sada knihoven umožňuje vytvářet komplexní webové aplikace v jazyce C# a Visual Basic .NET. Knihovna obsahuje již hotová řešení pro zabezpečení a šifrování, autentizaci uživatelů, správu formulářů a mnoho dalších.[4]

2.4 DevExtreme a DevExpress pro ASP.NET MVC

Jedná se o sadu tříd, nástrojů a komponent od společnosti Developer Express Inc., která slouží k vývoji responzivních webových aplikací. Sada DevExtreme poskytuje vizuální komponenty pro HTML5[6] a sada DevExpress poskytuje komponenty s rozšířenou funkcionalitou pro ASP.NET MVC.[5]

2.5 LINQ

LINQ je sada technik a rozšíření platformy .NET, které nám umožňují dotazování nad různými zdroji dat. Umožňuje nám například filtrování, řazení nebo propojování dat. Součástí je také automatické generování objektově-relačního mapování.[7]

2.6 PostgreSQL

PostgreSQL (zkráceně Postgres) je plnohodnotným relačním databázovým systémem s otevřeným zdrojovým kódem. Běží nativně na všech rozšířených operačních systémech včetně OS Linux, UNIX a Windows. Stoprocentně splňuje podmínky ACID, plně podporuje triggery, uložené procedury a spoustu dalších funkcí. Obsahuje většinu standardních datových typů. Nechybí ani podpora moderních datových typů jako je JSON nebo XML.[8]

2.7 Entity Framework a knihovna Npgsql

Entity Framework (EF) je sada technologií od společnost Microsoft, které podporují vývoj aplikací orientovaných na data. Jedná se o ORM (objektově-relační mapování), kdy se databázové tabulky přímo mapují na třídy v .NETu. V kódu pracujeme pouze s objekty a framework sám na pozadí generuje SQL dotazy. S jazykem SQL vůbec nepřijdeme do styku a naše aplikace je plně objektová[9]. Npgsql je knihovna s otevřeným zdrojovým kódem sloužící jako poskytovatel dat pro .NET aplikace, které díky němu mohou přistupovat do PostgreSQL databáze, podporuje také EF.[10]

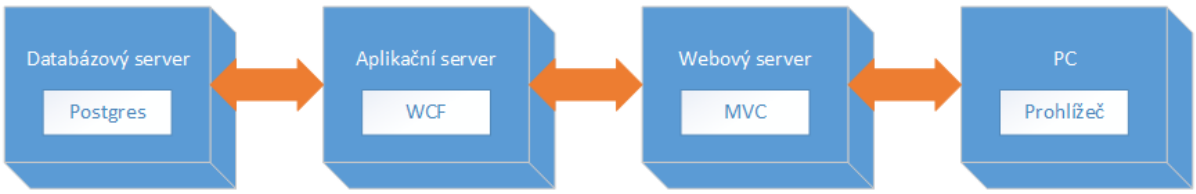
2.8 JavaScript a knihovny jQuery a OpenLayers

JavaScript (JS) je nenáročný, interpretovaný, objektově orientovaný jazyk, znám zejména jako skriptovací jazyk pro webové stránky.[11] Pro jednodušší programování v jazyce JavaScript byla vytvořena knihovna jQuery, která ulehčuje procházení struktury HTML dokumentů, práci s animacemi a událostmi, volání AJAX a mnoho dalšího.[12] Knihovna OpenLayers byla vyvinuta pro ulehčení práce s dynamickými mapami na webových stránkách. Umožňuje práci, mimo jiné, s mapovými vrstvami, mapovými vektorovými prvky a práci s různými geografickými informacemi.[13]

3 Architektura systému

Systém, na kterém jsem pracoval, se skládá ze tří serverů – databázového, aplikačního a webového. Jde o kombinaci třívrstvé architektury s technologií WCF (viz obr. 2).

Třívrstvá architektura se uplatňuje při přístupu k databázi – webový server, jakožto prezentační vrstva, nemůže přímo přistupovat k databázi, musí o to požádat WCF službu v aplikační vrstvě. MVC model se používá pro samotnou prezentaci dat uživateli webového prohlížeče a jeho komunikaci s webovým serverem.



Obrázek 2: Architektura systému

3.1 Databázový server

Databázový server je realizován SŘBD PostgreSQL. ORM je zajištěno pomocí rámce Entity Framework a knihovny Npgsql.

3.2 Aplikační server

Na aplikačním serveru běží většina výpočtů potřebných pro jednotlivé moduly. Aplikační server také realizuje volání metod pro přístup k databázi a business logiku. Aby tyto metody mohly být využity, běží na serveru webová WCF služba, která vystavuje rozhraní, pomocí kterého se lze dostat k metodám pro výpočty a přístup k databázi, které může využít každá aplikace, která se ke službě připojí.

3.3 Webový server

Na webovém serveru běží samotná webová aplikace GX GO. Server slouží hlavně k obsluze požadavků prohlížeče od klienta, případně také pro výpočet nenáročné logiky. Na serveru se využívá technologie ASP.NET MVC. Server komunikuje s několika službami, primárně s WCF službou běžící na aplikačním serveru, na kterou posílá požadavky o výpočty a práci s daty. Dále komunikuje také se službou pro výpočet tras, která běží na serveru superpočítačového centra IT4Innovations (dále jen IT4I routing), která umožňuje vyhledat v mapě trasu mezi dvěma zadanými souřadnicemi.

4 Úkoly zadané během praxe

V této kapitole jsou stručně popsány zadání jednotlivých úkolů odborné praxe. Jejich realizace je podrobně rozepsána v následující kapitole.

4.1 Opravy chyb v modulu routingu

Prvním úkolem, na který jsem byl nasazen, byly opravy a refaktorování modulu routingu v aplikaci GX GO. Pojmem refaktorování rozumíme proces provádění změn v programovém kódu takovým způsobem, který neovlivní vnější chování aplikace, ale vylepšuje jeho strukturu do přehlednější a čitelnější podoby. Modul routingu byl původně vyvíjen jiným studentem v rámci bakalářské práce. Způsob implementace bohužel nedodržoval zásady správného programování a bylo v něm zaneseno mnoho chyb. Mým úkolem tedy bylo identifikovat a opravit tyto chyby a uvést programový kód do správné podoby.

4.2 Uložení naplánované trasy

Dalším úkolem bylo rozšíření systému o možnost uložení naplánovaných tras do databáze a možnost jejich následného zobrazení, editace a mazání.

4.3 Odeslání naplánované trasy na email

V tomto úkolu mi bylo zadáno rozšířit modul routingu o funkcionalitu odeslání naplánované nebo uložené trasy na vybrané emaily z adresáře zákazníka. Požadavkem bylo odeslat email zahrnující výřez mapy s aktuálně zobrazenou trasou, itinerář s jednotlivými segmenty trasy, a také přílohu obsahující soubor s exportovanou trasou ve formátu KML.

4.4 Import/Export naplánované trasy

Tento úkol původně zahrnoval pouze požadavek na import trasy ze souboru formátu KML a její následné převedení do systému, kde by trasa byla přepočítána a zobrazena na mapě s možností další manipulace.

Během implementace byla zjištěna nedostatečná funkcionalita původního exportu tras, která neumožňovala exportovanou trasu bezproblémově importovat zpátky do systému a vznikla tedy motivace vylepšit export tras takovým způsobem, který standardizoval formát exportovaného souboru, a tedy možnost ho správně importovat zpět do systému.

4.5 Porovnání naplánované trasy s ujetou trasou

Posledním řešeným úkolem v rámci odborné praxe byl požadavek zákazníka o možnost porovnat uloženou trasu z databáze s ujetou trasou zvoleného vozidla. Detail požadavku tedy zněl, že poté, co vozidlo ujede určenou trasu, bude dostupná možnost porovnat původní uloženou trasu

s trasou, kterou vozidlo reálně ujelo. Tedy porovnání s jeho historickými pozicemi, a to s možností zobrazit odchylku, kdy se pozice ujeté trasy odchýlila od uložené trasy o více než uživatelem zadaný počet kilometrů.

5 Zvolené postupy řešení zadaných úkolů

5.1 Opravy chyb v modulu routingu

Prvním krokem tohoto úkolu bylo osvojit si funkčnost modulu routingu a analyzovat původní programový kód. Po splnění tohoto kroku nastala implementace oprav a změn konzultovaných s vedoucím.

První implementovanou změnou byl způsob, jakým webový klient komunikuje s webovým serverem. Tato komunikace probíhá pomocí technologie AJAX, kdy ze strany klienta jsou volány jednotlivé metody části Controller na straně webového serveru pomocí konstrukce `.ajax()` z knihovny jQuery a server poté vratí data parametrem zpět do funkce definované v konstrukci `.ajax()` (viz výpis 1).

```
$.ajax({
            type: "POST",
            url: "/Map/CalculateRoute",
            contentType: "application/json",
            data: JSON.stringify({ inputData: inputs }),
            success: function (callbackData) {
                showRoute(callbackData);
            }
});
```

Výpis 1: jQuery AJAX syntaxe

Tato data se předávají ve formátu JSON. V původním kódu byla všechna data tras (souřadnice, adresy bodů, názvy ulic atd.) skládána do dlouhých řetězců a spojována oddělovacími znaky, a následně naopak rozdělována do polí jak na straně klienta, tak na straně serveru. Toto řešení tvořilo velmi nepřehledný kód, ve kterém bylo obtížné odhalit chyby. Jako alternativa bylo implementováno objektové řešení, kdy se na straně serveru vytvořily modely pro data tras a na straně klienta objekty odpovídající předpisu těchto modelů. Původní složené řetězce byly zaměněny za tyto objekty. Výhodou ASP.NET MVC je technologie Model Binding, umožňující mimo jiné mapovat JSON objekty z klienta do objektů v jazyce C#. Tato technologie nám tedy umožnila vytvořit jednoduché a mnohem přehlednější řešení na obou stranách.

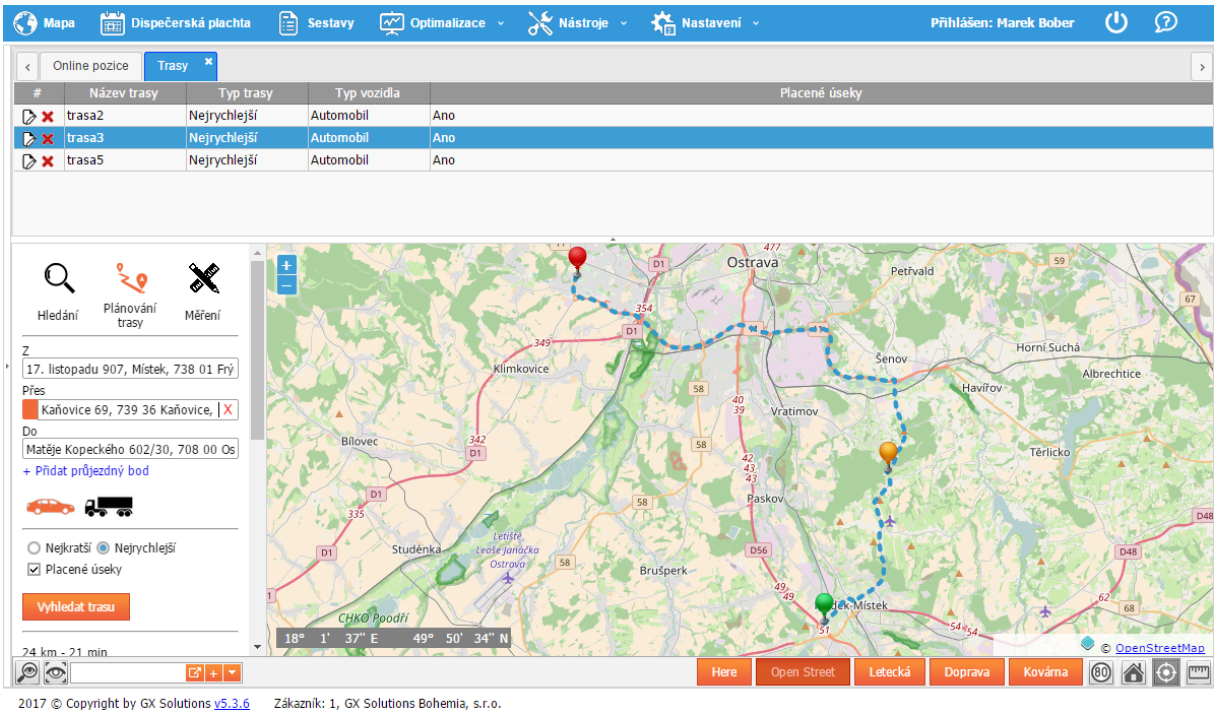
Druhou implementovanou změnou bylo ošetření chyb ve zpracování výsledku routingu, který vrací služba od IT4I, kdy byla webová aplikace za určitých vyhledávacích podmínek nestabilní. Na webovém serveru se nachází soubor s adresami ulic v České republice, webový server využívá tento soubor k mapování adres na ID ulic jednotlivých segmentů, které přicházejí ze služby od IT4I. Chyba nastávala za situace, kdy algoritmus procházející tyto segmenty nenašel v souboru a adresami dané ID ulice. Tento algoritmus byl řešený několika cykly, které pokrývaly jak případy jednoduchých tras, tak i složené trasy, které obsahují průjezdné body. Navržené řešení

zjednodušilo a zoptimalizovalo celý algoritmus na jeden cyklus, a zároveň pokrylo výjimky, které v něm mohou nastat.

Průběžně byly také implementovány drobné, ale četné změny jmenných konvencí, rozdělování funkcí do menších částí, doplňování chybějící dokumentace.

5.2 Uložení naplánované trasy

Tento úkol byl mým prvním plnohodnotným řešením, kdy došlo k rozšíření funkcionality na všech vrstvách systému, a tedy došlo i k rozšíření databáze o nové tabulky, byly implementovány nové metody v datové vrstvě a došlo k rozšíření WCF služby na straně aplikačního serveru.



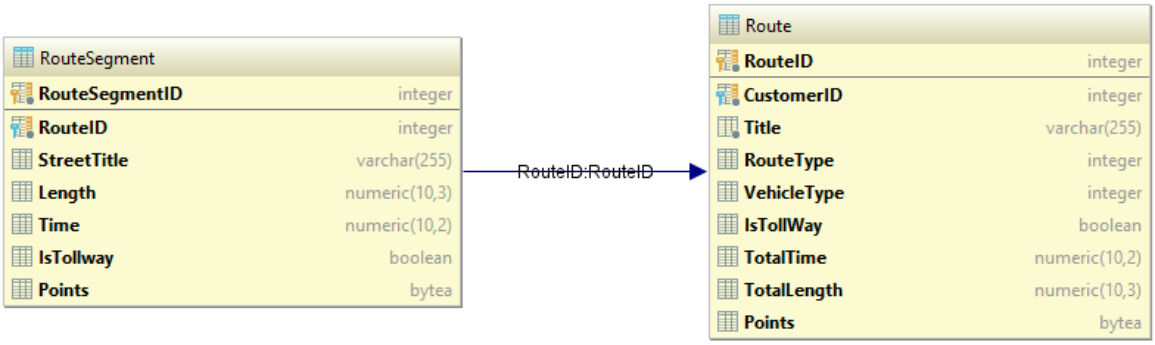
Obrázek 3: Tabulka s uloženými trasami a zobrazení trasy

Prvním krokem bylo vytvořit v GUI nová tlačítka pro uložení a smazání trasy, dále vytvořit formulář, kde uživatel zadá název trasy k uložení a implementace funkcionality těchto prvků, tedy vytvoření příslušných metod v části Controller. Dále bylo nutné přidat do mapových vrstev novou vrstvu pro zobrazení uložených tras. S úpravou souviselo také vytvoření nových prvků GUI nad mapou, které pomocí tabulkového zobrazení nabízí uživateli uložené trasy s možností zobrazení, editace a smazání. Jako zdroj této tabulky byl vytvořen nový model a příslušné metody na straně serveru. Důležitým krokem bylo zajistit vzájemné propojení funkcionality mezi levým panelem routingu a touto novou tabulkou tak, aby se při zvolení trasy vždy správně vyplnily data dané trasy (viz obr. 3). Dále bylo nutné zajistit správné načítání dat do pozadí webové aplikace tak, aby zobrazování jednotlivých tras bylo dostatečně rychlé. Řešením tedy bylo načítat do pozadí vždy všechny uložené trasy. V této chvíli implementace GUI běžela na lokálních testovacích

datech. Dalším krokem bylo tedy uložení tras ve fyzické PostgreSQL databázi. Byly navrženy tři nové tabulky – `Route`, `RouteSegment` a `RoutePoint`. Tabulka `Route` ukládala jednotlivé trasy, tabulka `RouteSegment` ukládala trasové segmenty (odpovídající většinou jedné ulici v trase) jednotlivých tras a tabulka `RoutePoint` ukládala jednotlivé souřadnicové body tvořící každý segment. V datové vrstvě byly pomocí technologie Entity Framework vytvořeny odpovídající základní CRUD metody, a také metoda vracející pole všech uložených tras zákazníka najednou. Následně byla rozšířena WCF služba na aplikační vrstvě o stejnou funkcionalitu a zajištěno mapování databázových objektů na DTO objekty pro komunikaci s webovým serverem. Webový server byl následně rozšířen o volání metod této WCF služby a předávání správných dat.

V této chvíli byla veškerá funkcionalita implementována, ale v testovací fázi nastal problém. Při pokusu o uložení delší trasy (např. napříč Českou republikou) trvalo samotné uložení i dvě minuty. Problémem byl velký počet jednotlivých souřadnicových bodů všech segmentů dané trasy. Tento počet byl rychlostním problémem jak pro Model Binder v MVC, který parsoval JSON data z klienta do DTO objektů k odeslání do aplikační vrstvy, tak pro samotnou databázi, která vytvářela ke každé trase desetitisíce záznamů.

Po analýze problému a konzultaci s vedoucím bylo přijato řešení zrušit tabulku `RoutePoint` a ukládat jednotlivé body segmentů pomocí bytových polí. Tabulka `RouteSegment` byla tedy rozšířena o sloupec typu `bytea` (datový typ pro bytové pole v PosgreSQL) a řešení úkolu bylo modifikováno pro práci s bytovými poli (viz obr. 4).



Obrázek 4: Výsledné schéma tabulek

Jednotlivé souřadnice byly ukládány v bytovém poli za sebou pomocí 32 bitových čísel s plovoucí desetinnou čárkou. Na straně webového serveru byla ke konverzi nově vypočítaných tras použitá .NET třída `BitConverter` a na straně klienta byla bytová pole konvertovaná pomocí tříd JavaScriptu `ArrayBuffer`, `Int8Array` a `Float32Array` určených k práci s byty. Problém s rychlostí parsování JSON dat Model Binderem v MVC byl vyřešen použitím knihovny Json.NET, která je v parsování velkých JSON objektů mnohem rychlejší. Toto řešení se ukázalo být správné, časy uložení tras se zkrátily s více než dvou minut na přibližně tři sekundy.

5.3 Odeslání naplánované trasy na email

Odesílání na email byl poměrně jednoduchý úkol, jehož cílem bylo naimplementovat v klientovi po vytvoření nebo zobrazení trasy tlačítko pro odeslání trasy na email. Toto tlačítko zobrazí formulář pro výběr emailu z adresáře uživatele. Úkol zahrnoval úpravy jak na straně klienta, tak na straně serveru. Pro odeslání dat z klienta na server, v tomto případě předmět emailu, obsah itineráře trasy, data KML souboru a bitová data obrázku s výřezem mapy, byl v MVC opět vytvořen model (viz výpis 2). Pro samotné odeslání emailu ze serveru byly použity standartní .NET třídy `Attachment`, `MailMessage` a `SmtpClient`.

```
public class RouteEmailModel
{
   public List<int> RecipientIDs { get; set; }
   public string EmailSubject { get; set; }
   public string KMLData { get; set; }
   public List<string> Segments { get; set; }
   public string ImageData { get; set; }
   public RouteEmailModel()
   {
       Segments = new List<string>();
   }
}
```

Výpis 2: MVC Model pro data emailu

Jediný problém v tomto úkolu byl s provedením mapového výřezu s trasou tak, aby na něm byla vidět vykreslená trasa. Knihovna JavaScriptu OpenLayers pro práci s mapovými poklady sice obsahuje funkce pro provádění výřezů, problém ale nastává při používání různých mapových podkladů (v našem případě Here Maps, OpenStreetMaps a satelitní mapy). Uživatel si totiž může podklady pod vykreslenou trasou libovolně přepínat a knihovna OpenLayers nezaznamená tyto změny. V systému již existovalo řešení tohoto problému, a to takovým způsobem, že pod uživateli viditelnou mapou se vytvoří nová, skrytá mapa, která kopíruje obsah viditelné mapy se zvoleným mapovým podkladem a prvky vykreslené trasy. Z této skryté mapy se poté udělá výřez. Toto řešení jsem použil a pouze upravil tak, aby fungovalo s mapovými prvky pro vykreslení trasy.

5.4 Import/Export naplánované trasy

Prvním krokem tohoto úkolu bylo osvojit si formát KML[14] do míry potřebné ke správnému parsování dat z tohoto formátu do formátu trasových dat v naší aplikaci, tzn. do formátu, který byl implementován v úkolu ukládání tras. Formát KML definovaný společností Google je aplikací jazyka XML, parsování tedy probíhá stejně jako u XML souborů. První implementace používala technologii LINQ to XML, což je rozšíření technologie LINQ pro dotazování XML

souborů podobným způsobem jako dotazování v relačních databázích. Pro naše účely, tzn. získání jednotlivých souřadnic bodů uložených v KML souboru, bylo potřeba procházet tagy s názvem Placemark a jejich potomky s názvem Point a Coordinates (viz výpis 3).

```
<?xml version="1.0" encoding="utf-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
   <name>Route1</name>
   <Folder>
      <Placemark>
         <name>Start</name>
         <snippet>Cajkovskeho</snippet>
         <Point>
            <coordinates>18.3357681981,49.6777233754</coordinates>
         </Point>
      </Placemark>
   </Folder>
</Document>
</kml>
```

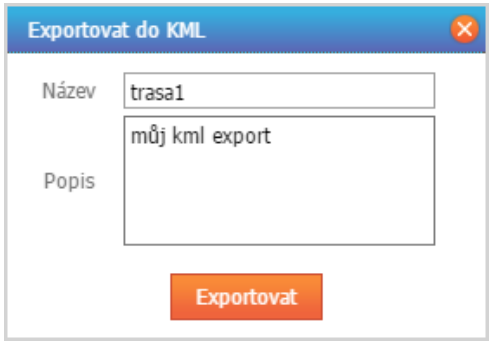
Výpis 3: Syntaxe KML

První problém tohoto úkolu vznikl při zjištění, že definice KML souborů je poměrně volná, tzn. tagy s názvem Placemark mohou být zanořené do dalších tagů, jako například Folder a tvořit i jiné konstrukce. Aplikace Earth společnosti Google například tvoří jinou strukturu KML než například export tras z webové aplikace Mapy.cz společnosti Seznam.cz. Řešení tedy vyžadovalo nástroj pro parsování všech druhů KML konstrukcí. Nakonec byla zvolena C# knihovna SharpKML, která se ukázala být efektivním nástrojem a došlo tedy k rozhodnutí přidat ji do naší aplikace.

Dalším problémem byl požadavek na přepočítání importované trasy službou od IT4I. Bylo nutné určit, jaké body z KML souboru použít pro definici průjezdných bodů v naší aplikaci. Aplikace Earth například v případě libovolné okružní trasy žádným způsobem neoznačuje průjezdné body této trasy (například města). Označuje pouze jednotlivé ulice a odbočení pro navigaci, kterých je ale příliš mnoho pro jejich použití v GUI. Řešením bylo použít jednotlivé body odbočení na trase jako průjezné body pro routing od IT4I, ale v GUI naší aplikace je skrýt. Výsledkem je tedy zachovaná okružní trasa, ale interaktivní zůstává pouze počáteční a koncový bod. Jiné aplikace jako například Mapy.cz průjezdné body označují i v KML souborech, takže zde nebyl problém je importovat i s průjezdnými body.

Při řešení přechozího problému byl zjištěn zásadní nedostatek v původním exportu trasy z naší aplikace (viz obr. 5), který ve výsledném souboru KML nedefinoval jak průjezdné body, tak ani body počáteční a cílové. Funkce exportu pracovala pouze s grafickými body pro vykreslení

trasové čáry na mapě. Kdyby tedy uživatel exportoval trasu s průjezdnými body, systém by ji nedokázal správně importovat zpět. Bylo tedy potřeba upravit původní export tras tak, aby sestavil takový KML soubor, který systém dokáže plně importovat zpět se všemi původními prvky.



Obrázek 5: Okno pro export trasy do KML

5.5 Porovnání naplánované trasy s ujetou trasou

Na začátku tohoto úkolu bylo rozhodnuto o přidání čtvrté záložky do panelu s nástroji, ve které bude uživatel poté, co zobrazí historii pozic určitého vozidla, vybírat trasu uloženou v databázi, se kterou se bude historie pozic porovnávat. Na záložce mělo být možné zadat také maximální odchylku v kilometrech, o kterou se můžou trasy lišit.

Prvním krokem bylo vytvořit tuto záložku, její obsah, tedy menu s výběrem uložených tras, textové pole s odchylkou, tlačítko pro porovnání a dále funkcionalitu porovnání implementovat. Zde bylo rozhodnuto, že ve výsledném porovnání tras by mělo být jasně vidět, kde se vozidlo odchýlilo od naplánované trasy, a to tak, že se v místě největší odchylky v daném úseku zobrazí čára mezi trasou vozidla a naplánovanou trasou s mírou odchylky v kilometrech.

Nejdůležitějším krokem bylo tedy implementovat algoritmus, který prochází jednotlivé body trasy vozidla a naplánované trasy a snaží se najít vždy dva nejvzdálenější korespondující body v daném úseku. Pokud je nalezne a body jsou od sebe dále, než je uživatelem zadaná maximální odchylka, vykreslí se pomocí knihovny OpenLayers mezi těmito dvěma body čára a vypíše se daná odchylka. Problémem bylo zajistit správné chování v případě nekorespondujících tras, tzn. pokud trasa byla delší nebo kratší než ujetá trasa vozidla. Kromě toho bylo nutné zajistit správnou funkcionalitu dalšího plánování tras. Byla proto vytvořena nová mapová vrstva pomocí knihovny OpenLayers a vykreslení tohoto porovnání bylo přesunuto do této vrstvy.

6 Shrnutí znalostí

6.1 Znalosti a dovednosti získané v průběhu studia uplatněné v průběhu odborné praxe

V průběhu mé odborné praxe jsem využil mnoho poznatků a vědomostí získaných během svého studia na Vysoké škole báňské. Hlavní znalosti mi daly předměty týkající se programování, jako například Programování I, II (PR I, II) a Algoritmy I, II (ALG I, II). Důležité pro mě byly také předměty týkající se programování na platformě .NET, tedy Programovací jazyky II (PJ II) a Architektura technologie .NET (AT .NET). Dále bych rád zmínil předmět Vývoj internetových aplikací (VIA), který mi poskytl mnoho potřebných znalosti tvorby webových aplikací a předměty Databázové a informační systémy (DAIS) a Vývoj informačních systémů (VIS), které mi poskytly základy práce s databázemi a vývoje vícevrstvých aplikací.

6.2 Znalosti a dovednosti scházející v průběhu odborné praxe

Již v prvním týdnu mé odborné praxe jsem dospěl k závěru, že znalostí mi chybí nespočet. Toto mě po celou dobu odborné praxe nutilo učit se novým věcem a studovat nové technologie.

Hlavní zkušeností, která mi scházela, byl praktický vývoj aplikací v týmu, tedy nejen týmová spolupráce, ale také zkušenosti s použitím softwaru pro správu a verzování zdrojového kódu, v našem případě aplikace Apache Subversion. Dále bych rád zmínil značný problém, kdy jsem jako student, který pracoval pouze na relativně malých školních projektech, přešel k firemní vícevrstvé aplikaci, která obsahuje stovky databázových tabulek a tisíce tříd. Zde mi chyběly zkušenosti s orientací v obsáhlých reálných aplikacích.

Další chybějící znalostí byly reálné praktické postupy implementace konkrétních technologií. Předměty jako Architektura technologie .NET (AT .NET) nebo Vývoj internetových aplikací (VIA) mi daly znalosti mnoha technologií, ale ne do takové hloubky, která je potřebná pro jejich praktickou aplikaci. Konkrétně mi scházely hlubší znalosti s vývojem technologií WCF a tzv. jedno-stránkových webových aplikací (Single-page application, SPA).

V neposlední řadě jsem se musel doučit detaily databázového systému PostgreSQL, i když v tomto případě se jednalo spíše o drobné věci, jako například nestandartní jména datových typů. Základy tohoto SŘBD jsou podobné jako to, s čím jsem se seznámil během mých studií.

7 Závěr

Během odborné praxe jsem měl možnost rozšířit komplexní webovou aplikaci pro sledování vozidel o novou funkcionalitu. Vyzkoušel jsem si také, jak vypadá práce v moderní firmě a jak pracovat na týmových projektech. Měl jsem možnost poznat, jak vypadají, fungují a jak se vytvářejí aplikace pro telematické systémy, nejen ze strany software, ale také hardware. Došlo také k velkému posunu mých programovacích schopností, jelikož jsem měl stálou možnost konzultace s ostatními kolegy z oddělení, kteří mi ochotně poradili vhodné postupy.

Ve výsledku tedy hodnotím mou odbornou praxi kladně a jsem rád za tuto zkušenost, která byla posunem v mém profesním vývoji a zajímavým a praktickým zakončením mého bakalářského studia.

Literatura

[1] *O firmě GX Solutions* [online]. 2017 [cit. 2017-04-04]. Dostupné z: http://gxsolutions.cz/o_firme/

[2] *Úvod do .NET Frameworku* [online]. 2017 [cit. 2017-04-04]. Dostupné z: http://www.dotnetportal.cz/clanek/125/Uvod-do-NET-Frameworku

[3] *MSDN Windows Communication Foundation* [online]. 2017 [cit. 2017-04-04]. Dostupné z: https://msdn.microsoft.com/en-us/library/dd456779%28v=vs.110%29.aspx

[4] *MSDN ASP.NET MVC Overview* [online]. 2017 [cit. 2017-04-04]. Dostupné z: https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx

[5] *ASP.NET Controls and MVC Extensions* [online]. 2017 [cit. 2017-04-04]. Dostupné z: https://documentation.devexpress.com/#AspNet/CustomDocument7873

[6] *Introduction to DevExtreme* [online]. 2017 [cit. 2017-04-04]. Dostupné z: https://js.devexpress.com/Documentation/16_2/Guide/Common/Introduction_to_DevExtreme/

[7] *LINQ a EF - Úvod* [online]. 2017 [cit. 2017-04-04]. Dostupné z: http://www.dotnetportal.cz/clanek/6413/LINQ-a-EF-Uvod

[8] *About PostgreSQL* [online]. 2017 [cit. 2017-04-04]. Dostupné z: https://www.postgresql.org/about/

[9] *Scaffolding a Entity Framework v ASP.NET MVC* [online]. 2017 [cit. 2017-04-04]. Dostupné z: http://www.itnetwork.cz/csharp/asp-net/mvc/asp-dot-net-mvc-tutorial-scaffolding-entity-framework-editor-clanku

[10] *Npgsql - .NET Access to PostgreSQL* [online]. 2017 [cit. 2017-04-04]. Dostupné z: http://www.npgsql.org/

[11] *MDN JavaScript* [online]. 2017 [cit. 2017-04-04]. Dostupné z: https://developer.mozilla.org/cs/docs/Web/JavaScript

[12] *About jQuery* [online]. 2017 [cit. 2017-04-04]. Dostupné z: https://jquery.com/

[13] *OpenLayers* [online]. 2017 [cit. 2017-04-04]. Dostupné z: http://openlayers.org/

[14] *KML Reference* [online]. 2017 [cit. 2017-04-04]. Dostupné z: https://developers.google.com/kml/documentation/kmlreference